EVIM

ENGLISH VALIDATION
&
INTERNSHIP MANAGEMENT

Unit Test Plan

EVIM - English Validation & Internship Management

Riferimento	
Versione	1.0
Data	24/12/2019
Destinatario	Top Management
Presentato da	Simone Auriemma - Michele Duraccio – Antonio Giano – Simona Grieco – Emilio Schiavo – Maria Concetta Schiavone – Nicola Sisti – Vincenzo Colacicco
Approvato da	Edoardo Carpentiero – Attilio Della Greca

Team Composition

Ruolo	Nome	Posizione	Contatti
Top Manager	Filomena Ferrucci	Rappresentante del client	f.ferrucci@unisa.it
Project Manager	Edoardo Carpentiero	Project Manager	e.carpentiero1@studenti.unisa.it
Project Manager	Attilio Della Greca	Project Manager	a.dellagreca5@studenti.unisa.it
Team Member	Simone Auriemma		s.auriemma5@studenti.unisa.it
Team Member	Vincenzo Colacicco		v.colacicco1@studenti.unisa.it
Team Member	Duraccio Michele		m.duraccio3@studenti.unisa.it
Team Member	Giano Antonio		a.giano1@studenti.unisa.it
Team Member	Grieco Simona		s.grieco13@studenti.unisa.it
Team Member	Emilio Schiavo		e.schiavo8@studenti.unisa.it
Team Member	Maria Concetta Schiavone		m.schiavone29@studenti.unisa.it
Team Member	Nicola Sisti		n.sisti1@studenti.unisa.it

Sommario

Revision History

Data	Versione	Descrizione	Autori
22/12/2019	0.1	Impostazione documento	Michele Duraccio Vincenzo Colacicco
24/12/2019	0.2	Aggiornamento classi da testare	Michele Duraccio

1. Introduzione

1.1 Scopo del sistema

Il Sistema Internship Management, integrato con la piattaforma English Validation, si pone come obiettivo principale la digitalizzazione di tutte le pratiche necessarie per lo svolgimento del Tirocinio formativo o il riconoscimento di attività lavorativa svolta, in modo da superare definitivamente i costi e le inefficienze della gestione cartacea del processo, garantendo una gestione decentralizzata ed efficace, così da avere ogni documento disponibile in rete ed accessibile alle parti interessate da qualsiasi luogo ed in qualsiasi momento.

1.2 Scopo del documento

Il testing di unità rappresenta la fase di testing in cui si assicura che le componenti sviluppate funzionino in isolamento.

Questo documento ha il compito di identificare e pianificare la strategia di testing di unità per il sistema EVIM.

1.3 Riferimenti

- Kathy Schwalbe, "Information Technology Project Management", International Edition 7E, Cengage Learning, 2014;
- Bernd Bruegge, Allen H. Dutoit, "Object-Oriented Software Engineering Using UML, Patterns and Java", Third Ed., Pearson, 2010;
- EVIM_TP_Vers.1.1
- EVIM_ODD_Vers.1.1
- EVIM_RAD_Vers.1.5
- EVIM_TCS_Vers.1.2

2. Unit Test Plan - Model

Di seguito sono riportate le componenti da testare presenti nel Package Model in ordine di priorità. Il testing viene effettuato attraverso JUnit.

Componenti
DriverManagerConnectionPool
User – User DAO
Riconoscimento – RiconoscimentoDAO
TirocinioInterno - TirocinioInternoDAO
TirocinioEsterno - TirocinioEsternoDAO
TutorAccademico – TutorAccademicoDAO
TutorAziendale - TutorAziendaleDAO
Azienda - AziendaDAO
Proposta - PropostaDAO
ReferenteAziendale - ReferenteAziendaleDAO
Convenzione - ConvenzioneDAO
Attività - AttivitàDAO
Registro
PDFProgettoFormativo

3. Unit Test Plan - Controller

Di seguito sono riportate le componenti da testare presenti nel Package Controller in ordine di priorità. Il testing viene effettuato attraverso Mockito.

Componenti da testare
GestioneAutenticazione
GestioneAccount
GestioneProposta Tirocinio
GestioneRichiestaTirocinio
GestioneTirocinio
GestioneRegistroTirocinio
GestioneRiconoscimentoAttività

4. Pass/Fail Criteria

Il testing avrà successo se l'output osservato è diverso dall'output atteso, che ci viene dettato dall'oracolo. Questo significa che la fase di testing avrà successo se individuerà una failure, ovvero un comportamento che si manifesta quando il servizio offerto dal sistema devia dalle specifiche.

In tal caso questa verrà analizzata e, se legata ad un fault, si procederà alla sua correzione. Sarà infine iterata la fase di testing per verificare che la modifica non abbia impattato su altre componenti del sistema.

Al contrario, il testing fallirà se l'output osservato sarà uguale a quello dettato dall'oracolo.

Glossario

Mockito: framework di test open source per java

JUnit: è un framework di unit testing per il linguaggio di programmazione Java.